510-63
188130
130
188130

Maximum Likelihood Estimation of Parameterized 3-D Surfaces Using a Moving Camera

Y. Hung, B. Cernuschi-Frias, and D.B. Cooper
Brown University
Providence, RI 02912

B 172031

Abstract

A new approach is introduced to estimating object surfaces in three-dimensional space from a sequence of images. A surface of interest here is modeled as a 3-D function known up to the values of a few parameters. The approach will work with any parameterization. However, in work to date we have modeled objects as patches of spheres, cylinders, and planes,---primitive objects. These primitive surfaces are special cases of 3-D quadric surfaces. Primitive surface estimation is treated as the general problem of **maximum likelihood parameter estimation** based on two or more functionally related data sets. In our case, these data sets constitute a sequence of images taken at different locations and orientations. A simple geometric explanation is given for the estimation algorithm. Though various techniques can be used to implement this nonlinear estimation, we discuss the use of gradient descent. Experiments are run and discussed for the case of a sphere of unknown location. These experiments graphically illustrate the various advantages of using as many images as possible in the estimation and of distributing camera positions from first to last over as large a baseline as possible. In order to extract all the usable information from the sequence of images, all the images should be available simultaneously for the parameter estimation. We introduce the use of asymptotic Bayesian approximations in order to summarize the useful information in a sequence of images, thereby drastically reducing both the storage and amount of processing required. The attractiveness of our Bayesian approach is that now all the usual tools of statistical signal analysis can be brought to bear, the information extraction appears to be robust and computationally reasonable, the concepts are geometric and simple, and essentially optimal accuracy should result.

I. Introduction

Essentially all 3-D object surface estimation from multiple views to date is based on either active stereo using a laser and one or two cameras for triangulation, or on passive stereo involving matching points in two images and using triangulation, or on optical flow [1], [10], [11]. We suggest a new approach in which surfaces of complex objects are approximated by a few patches of 3-D parameterized surfaces, and these parameters are estimated from two or more images taken by calibrated cameras from different locations and directions. These parameterized patches are referred to as *primitive objects*. We formulate the parameter estimation problem as standard maximum likelihood estimation, given two or more functionally related data sets. Estimation accuracy is achieved by processing data in blocks (which may be large), in addition to processing many images and with camera positions distributed over as large a baseline as possible. The actual processing is simple standard statistical signal analysis. This approach, first presented in [4], is completely new as far as we know. In summary, the contribution of this paper is the treatment of 3-D surface inference as a standard *maximum likelihood parameter estimation problem* requiring low data storage capacity and where parameter estimates are updated recursively as each new image in a sequence of images is received and processed.

Central to 3-D surface estimation from two (or more) images taken from cameras in different locations and orientations is the pairing of points from two images that are images of the same point on a 3-D surface. This matching of points in two images is usually done in either of two ways. (i) If the two cameras are physically close and their optical axes are almost parallel, then their images will differ from one another only by translation---one will be a shifted version of the other. Then image 1 can be partitioned into patches, and each patch cross-correlated with image 2 to find its location in image 2. Once this correspondence is known, the location of the surface region in 3-D space seen in the pair of corresponding image patches can be determined by triangulation. Since the surface region seen is usually curved, one would like the patches to be small in order to locate the surface region seen accurately. However, if the images are noisy, large surface patches must be used to accurately estimate a pair of corresponding patches in the two images.

Significant triangulation errors occur when the camera optical axes are close together and almost parallel because of matching errors due to image noise, and because 3-D object surfaces are curved. Additional triangulation error occurs because there is some error in camera calibration. (ii) An alternative approach that permits a large angle between the camera optical axes to improve triangulation accuracy is to locate corresponding small local features in the two images. An example of such a feature is a vertex of a polyhedron. For a curved surface, contours on the surface are features often used to be matched in pairs of images. The difficulty here is that a large amount of pattern recognition may be necessary to recognize a pair of corresponding features in the two images. Past efforts at cross-correlation of large image patches, as in (i), has been unsuccessful here because a patch in one image will be a distorted version of a corresponding patch in the other image.

The work closest in spirit to ours is the recent work of Faugeras, Ayache and Faverjon [8], who develop the idea of estimating points and lines on a 3-D object surface, or planar surfaces, from a sequence of images. More specifically, they assume that the probability distribution for the estimates of points on a surface based on a pair of images is known. They then assume that a sequence of such estimates and associated distributions are known for a sequence of images. Their contribution, then, is to use the extended Kalman filter for combining this sequence of estimates to obtain improved estimates of the surface points. They derive the equations for estimating lines, and suggest that it can be extended to planes. Among the errors they take into account, are those in camera calibration. Their concept is important, though they do not tackle here the problem of optimally estimating the surface points or lines directly from the data in the images.

Our paper is an expansion of one where our 3-D surface estimation algorithm was first proposed [4]. In subsequent papers, we showed that our basic estimation algorithm is maximum likelihood estimation, and derived Cramer-Rao *irreducible* lower bounds on the parameter estimation error covariance matrices [6], and we also discussed the use of Markov Random Field (stochastic process) models for 3-D surfaces [5] as a generalization of the use of parameterized surface models. These and the present paper together constitute a new Bayesian theory for 3-D surface estimation based on a sequence of noisy images.

Sections II.A - II.C introduce the transformations necessary for understanding the relation of images in two or more views. Sections III.A - III.B describe the performance functional and the gradient descent algorithm used in estimating the a priori unknown 3-D object parameters based on the use of two images. Section III.C provides a very simple geometric interpretation of the algorithm. Sections IV.A - IV.D extend the approach for use of a sequence of images that might be taken by a moving camera. In order to arrive at a computationally feasible algorithm, we introduce the use of maximum likelihood estimation here. This development also points out that the algorithm described in section III is maximum likelihood estimation. The importance of this observations is that maximum likelihood estimators are known to converge to the true parameter values, and are known to have minimum estimation error covariance as the number of observations become large. In section V we introduce a somewhat different estimator for a moving camera, and point out that it has certain desirable computational properties but is *less accurate*. This algorithm is somewhat similar to the use of *optical flow*.

II.A Notation and Description of Camera Motion

Let P be a point in 3-D space and $\mathbf{r} = (x \;\; y \;\; z)^{T\dagger}$ *be its coordinates in the fixed orthogonal world reference frame.* Since we assume that objects do not move, this reference frame is fixed with respect to the objects viewed by the camera, and we will call it the object reference frame (ORF). Let $\mathbf{r}(n) = (x_n \;\; y_n \;\; z_n)^T$ be the coordinates of the point P in CRFn, the reference frame attached to camera n. This reference frame is such that: (1) the camera optical axis is parallel to the z_n axis, and it looks at the negative z_n axis; (2) the x_n and y_n axes are parallel to the sides of the image; (3) the origin of the reference frame coincides with the center of the image plane. The image is corrected so that the view is not inverted top to bottom and left to right, i.e., a central projection is used.

Let B(n) denote the 3×3 orthogonal rotation matrix that specifies the three unit coordinate vectors for CRFn in terms of the three unit coordinate vectors for the ORF. Let $\mathbf{r}_c(n)$ specify the origin of CRFn in the ORF. Then

$$\mathbf{r}(n) = B^T(n)\left[\mathbf{r} - \mathbf{r}_c(n)\right], \qquad \text{and} \qquad \mathbf{r} = B(n)\mathbf{r}(n) + \mathbf{r}_c(n). \tag{1}$$

The rotation matrix B(n) and the translation vector $\mathbf{r}_c(n)$ are known for calibrated cameras. In this paper, we will use $\mathbf{b}_n$ to represent a vector having as its components the parameters that specify both B(n) and $\mathbf{r}_c(n)$.

† A symbol in boldface is a column vector, a superscript capital T attached to a vector denotes vector transpose.

II.B Surface Parameterization

Our approach is applicable to *any parameterized surface*. A few researchers have used differential geometric properties, such as Gaussian curvature and mean curvature, to describe surfaces, see [2]. These are useful for surface parameterization because they are coordinate free. In general, the surfaces we want to estimate can be described by an implicit function with respect to the ORF:

$$g(r; a) = g(x, y, z; a) = 0, \tag{2}$$

where a is the parameters describing the surface with respect to the ORF. For example, the equation for the general quadratic surface is

$$a_{11}x^2 + 2a_{12}xy + 2a_{13}xz + a_{22}y^2 + 2a_{23}yz + a_{33}z^2 + 2a_{14}x + 2a_{24}y + 2a_{34}z + a_{44} = 0. \tag{3}$$

In this case we denote $a = (a_{11}, a_{12}, \ldots, a_{44})^T$.

II.C Images of an Object Surface Point in Two Image Frames

As shown in Fig. 1.a, P denotes a point on a parameterized 3-D surface of interest. This surface is described by a function in the ORF (see section II.B). The function is uniquely determined by specifying the values of a parameter vector a. Point P on the object surface is seen as points having coordinates s and u in images 1 and 2, respectively. We assume a *Lambertian reflectance model*. Then the images of point P at s and u will have the same intensity. The techniques proposed will not apply to specular reflectors, without modification, because the location of points on the object surface at which specular reflection occurs depends on the camera location. Since most surfaces of interest are largely Lambertian, the assumption is a useful one. Hence,

$$I_2(u) = I_1(s) \tag{4}$$

where $I_1(u)$ and $I_2(s)$ are the picture functions (image intensity functions) in Frames 1 and 2, respectively. For those cases where the Lambertian assumption does not apply, a possible modified approach is to use an edge map. Here, pixels are given values of 128 or 0 depending on whether they are detected as being edge points or non edge points, respectively. These maps are then smoothed to obtain more continuous arrays, and these are used as though they are regular picture functions in our estimation algorithms. The usefulness of the edge map is that it is a representation of rapid changes in the object surface patterns, and largely unaffected by the presence of some specular component in the object surface. Experiments using edge maps with our algorithm are described in [6].

For simplicity, we use the orthographic projection model [7] for image formation, i.e., all rays from points on the object surface to the camera are roughly parallel. (With slight modification, all of our results can be used with the perspective projection model.) Let $r(1) = (x_1\ y_1\ z_1)^T$ be the coordinates of the 3-D surface point P with respect to CRF1, and $r(2) = (x_2\ y_2\ z_2)^T$ be the coordinates of the point P with respect to CRF2. Then, under the assumption of orthographic projection,

$$s = (x_1\ \ y_1)^T, \qquad u = (x_2\ \ y_2)^T.$$

If we pick a point s in image plane 1, it will correspond to some point P on the 3-D surface. If this point P is also seen in image 2, its image in image plane 2 will occur at some coordinate u. Therefore, given some point s in image 1, if we want to compute the corresponding image point u in image 2 based on the current estimation of a, we can:
(i) first, find the 3-D location of the corresponding surface point P;
(ii) then, find the image point u corresponding to P.

In step (i), represent the surface point P with respect to CRF1 by $r(1) = (x_1\ \ y_1\ \ z_1)^T$. Using equations (1) and (2), the equation of the surface is

$$g(r;\ a) = g(\ B(1)r(1) + r_c(1)\ ;\ a) = g(\ B(1)(x_1\ y_1\ z_1)^T + r_c(1)\ ;\ a) = 0. \tag{5}$$

Since the point P resides on the surface, r(1) must satisfy the above equation. Therefore, given $s = (x_1\ \ y_1)^T$, we can solve equation (5) for z_1. An example for the spherical surface is given in the next section.

In step (ii), we want to compute u. Now that we have obtained r(1) from step (i), using equation (1) we can compute $u = (x_2\ \ y_2)^T$ by

$$r(2) = (x_2 \;\; y_2 \;\; z_2)^T = B^T(2)\left[r - r_c(1)\right] = B^T(2)\,B(1)\,r(1) + B^T(2)\left[r_c(1) - r_c(2)\right]. \tag{6}$$

Call $C = B^T(2)B(1)$ and $d = B^T(2)\;(r_c(1) - r_c(2))$. Then, partition the C matrix and d vector as:

$$C = \begin{bmatrix} C_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}, \qquad d = \begin{bmatrix} e \\ d_3 \end{bmatrix},$$

where C_{11} is 2×2, c_{12} and c_{21}^T are 2×1, c_{22} is a number, e is 2×1, and d_3 is a number. From the preceding:

$$u = C_{11}s + c_{12}z_1(s,b_1,a) + e. \tag{6a}$$

Combining steps (i) and (ii) above, we denote the functional relationship (6a) between s and u by

$$u = h(s,\, b,\, z(s,a)\,), \tag{7}$$

where the vector b includes b_1 and b_2, and specifies C_{11}, c_{12}, and e.

III.A Estimation of the Parameterized Surface Using Two Images

If we know the camera position, b, and the true surface parameters, a_T, then

$$I_1(s) = I_2(\,h(s,b,z(s,a_T))\,) \tag{8}$$

for each s. Choose a region in image 1. Denote this pixel set in this region by D. Consider the error measure

$$e_D(a) = \sum_{s\in D}\left[I_1(s) - I_2(\,h(s,b,z(s,a))\,)\right]^2. \tag{9}$$

Then $e_D(a)$ is a minimum at $a = a_T$. Our problem is to estimate a_T by minimizing (9) with respect to a.

To estimate a_T that minimizes (9), we choose to use the gradient method as follows:

$$a_{n+1} = a_n - \frac{\partial e_D(a_n)}{\partial a}\Delta_n, \tag{10}$$

where Δ_n depends on $e_D(a_n)$ and $\dfrac{\partial e_D(a_n)}{\partial a}$ and has magnitude that goes to 0 as n goes to infinity.

There are several ways to compute the gradient $\dfrac{\partial e_D}{\partial a}$. We present one of the methods used in our experiments. Taking the derivative of (9) with respect to a, we have

$$\frac{\partial e_D}{\partial a} = -2\sum_{s\in D}\left[I_1(s) - I_2(u)\right]\frac{\partial I_2(u)}{\partial a}, \tag{11}$$

where u is a function of a as shown in (7). Use of the chain rule gives‡

$$\frac{\partial I_2(u)}{\partial a} = \frac{\partial I_2(u)}{\partial u}\,\frac{\partial u}{\partial z}\,\frac{\partial z(s,\,a)}{\partial a}, \tag{12}$$

where $u = h(s,\, b,\, z(s,\, a)\,)$ as in equation (7). The first term $\dfrac{\partial I_2(u)}{\partial u}$ can be computed approximately using the sobel operator. The second term $\dfrac{\partial u}{\partial z}$ is just a constant provided that we assume the orthographic projection model. This can be shown as follows. From equation (1)

$$r(2) = B^T(2)\left[r - r_c(2)\right], \tag{1}$$

and upon using the notation

$$r(2) = (x_2 \;\; y_2 \;\; z_2)^T, \qquad u = (x_2 \;\; y_2)^T, \qquad r = (x \;\; y \;\; z)^T,$$

‡ The notation used here is that $\dfrac{\partial I_2(u)}{\partial a}$ is a K component row vector, where K is the number of the components in column vector a.

we have $\frac{\partial u}{\partial z} = (B^T(2)_{13}\ \ B^T(2)_{23})^T$, where $B^T(2)_{ij}$ means the ijth element of matrix $B^T(2)$.

In general, it may be inconvenient to express z as an explicit function of a. Hence, we compute the third term by

$$\frac{\partial z(s,\ a)}{\partial a} = -\frac{\partial g(x,y,z,a)}{\partial a} \Big/ \frac{\partial g(x,y,z,a)}{\partial z}. \tag{13}$$

III.B An Example: The Sphere

To illustrate the approach, consider a spherical surface described by the equation

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = R^2. \tag{14}$$

For this surface, z can be solved for explicitly, via

$$z = z_0 \pm (R^2 - (x - x_o)^2 - (y - y_o)^2)^{1/2}. \tag{15}$$

The positive square root is used since the outside surface of the sphere is seen by the camera looking in the negative z direction. Hence,

$$\begin{aligned}\left(\frac{\partial z(s,\ a)}{\partial a}\right) &= \left(\frac{\partial z}{\partial x_o}\ \ \frac{\partial z}{\partial y_o}\ \ \frac{\partial z}{\partial z_o}\ \ \frac{\partial z}{\partial R}\right) \\ &= \left(\ (x - x_o)/(z - z_o),\ \ (y - y_o)/(z - z_o),\ \ 1,\ \ R/(z - z_o)\ \right)\end{aligned} \tag{16}$$

and $z - z_o = (R^2 - (x - x_o)^2 - (y - y_o)^2\ )^{1/2}$. The vector $\frac{\partial z}{\partial a}$ can be computed directly from this.

The analogous equations for planes, cylinder and general quadrics are presented in [6].

III.C Algorithm Operation Interpretation

Fig. 1b is useful for illustrating, in two dimensions, the operation of our algorithm for estimating a_t. Spheres in 3-D are shown as circles. Consider the processing of the image patch between points s′ and s″ in Frame 1. This patch is the image of the patch between points p′ and p″ on the true sphere labeled a_t. The same patch on the sphere surface gives rise to the image patch between points u′ and u″ in Frame 2. Now suppose the system's estimation of a_t is $\hat{a}$. The associated sphere is shown. The performance functional for the estimate of a is given by (9) and is computed as follows. The system thinks that the locations on the sphere surface that give rise to the images at points s′ and s″ in Frame 1 are the intersections of the dashed lines, from s′ and s″, with the sphere labeled $\hat{a}$. These sphere surface points would be seen as the images at point $\hat{u}'$ and $\hat{u}''$ in Frame 2. Hence, the system takes the image patch between points $\hat{u}'$ and $\hat{u}''$ in Frame 2 and assumes that the image at each point u in this interval is the same image as the image at a point s in the interval between s′ and s″ in Frame 1. The points u and s are related geometrically as in the figure, or algebraically by (4). Performance functional (9) requires computing this error $I_1(s) - I_2(\ h(s, b, z(s,a))\ )$.

We make the following interesting observations. From the geometry of image formation in Fig. 1b, the varying scale change that maps the image patch over interval [s′, s″] in Frame 1 into the image patch over interval [u′, u″] is seen. Note that both a scale change and a translation are involved in this 2-D illustration.

If the incorrect $\hat{a}$ is used in computing the performance functional (9), the patch of image used in Frame 2 is that over the interval [$\hat{u}'$, $\hat{u}''$]. Note that this interval is both a shift and a varying scaling of the interval [u′, u″]. If instead of a sphere, we were dealing with a planar surface, the scale change would be constant throughout the image.

IV Estimation of Parametrized Surface Based on a Sequence of Images

Now suppose a sequence of images is available for estimating a_T, the true parameters of the surface. How can best use be made of this data set? In this section we develop a computationally reasonable approximately maximum likelihood estimator (mle) for a_T.

IV.A The Model

The model that we use for the nth image $I_n(u)$, $u \in D_n$, is that of some true picture function $\mu_n(u)$ plus additive white noise having variance σ^2. Hence, $I_n(u)$, $u \in D_n$, is a set of random variables having joint probability density

function (pdf)

$$(2\pi\sigma^2)^{-d_n/2} \exp \sum_{u \in D_n} \left\{ -(1/2\sigma^2) \left[I_n(u) - \mu_n(u) \right]^2 \right\} \tag{17}$$

where d_n is the number of pixels in D_n. We introduce the more compact notation: $u_n(s,a) = h(s,b_n,a)$, and $s_n(u,a) = h^{-1}(u,b_n,a)$, where b_n is the transformation parameters specifying the n^{th} camera position. Let I_n denote the vector of picture function values, i.e., it has components $I_n(u)$, $u \in D_n$. Let μ_n denote the vector having components $\mu_n(u)$, $u \in D_n$. Then (17) is a function of the parameter vector $(\mu_n^T \ \sigma^2 \ a^T)^T$. Because of the Lambertian assumption for image formation,

$$\mu_n(u) = \mu_1(s_n(u,a)). \tag{18}$$

Hence, the μ_n for all n can be specified in terms of μ_1. Then $\alpha = (\mu_1^T \ \sigma^2 \ a^T)^T$ is a parameter vector that specifies the pdf's (17), for all I_n. Since the additive image noise is independent from image to image, the log of the joint pdf of $I_1, \ldots, I_N$ is

$$\begin{aligned} L_N(\alpha) &= \ln p(I_1, I_2, \ldots, I_N \mid \alpha) \\ &= -\left[\sum_{n=1}^{N} d_n/2 \right] \left[\ln 2\pi\sigma^2 \right] - (1/2\sigma^2) \sum_{n=1}^{N} \left\{ \sum_{u \in D_n} \left[I_n(u) - \mu_1(s_n(u,a)) \right]^2 \right\}. \end{aligned} \tag{19}$$

Our goal is to find $\hat{a}_N$ that maximizes (19). Since this estimate is a maximum likelihood estimate (mle), we know that it has certain desirable properties such as converging to a_T as $\sum_{n=1}^{N} d_n \to \infty$, and having minimum covariance matrix for the error in the estimation of a_T as $\sum_{n=1}^{N} d_n$ becomes large. The difficulty here is that μ_1 is a priori unknown. Hence, in order to compute $\hat{a}_N$ we must simultaneously compute $\hat{\mu}_{1N}$, the estimate of μ_1 based on Isun1, $\ldots, I_N$. Though this looks like a formidable computational challenge, it is in fact easily manageable. In [6] we showed that (9), the performance functional we minimize for estimating a_T in the two picture case, is equivalent to (19) for N=2.

IV.B The Asymptotic Representation

As in section III.A, gradient methods can be used for minimizing $-L_N(\alpha)$. A problem here is that N images must be stored and processed simultaneously. This incurs both a great amount of storage and a large amount of processing for each N. An effective approximation for avoiding this storage problem can be had as follows. Let $\underline{I}_N$ denote $I_1, I_2, \ldots, I_N$. In [3] it is shown that

$$p(\underline{I}_N \mid \alpha) \approx p(\underline{I}_N \mid \hat{\alpha}_N) \exp\left\{ -(1/2)(\alpha - \hat{\alpha}_N)^T \Psi(\underline{I}_N, \hat{\alpha}_N)(\alpha - \hat{\alpha}_N) \right\} \tag{20}$$

where the function $\Psi(\underline{I}_N, \hat{\alpha}_n)$ is a K×K matrix having ijth element

$$[\Psi(\underline{I}_N, \hat{\alpha}_N)]_{ij} = -\frac{\partial^2}{\partial\alpha_j \partial\alpha_i} \ln p(\underline{I}_N \mid \alpha) \Big|_{\alpha = \hat{\alpha}_N} \tag{21}$$

and K is the number of components in α. Hence, (20) has a Gaussian shape in α with mean α_N and inverse covariance matrix $\Psi(\underline{I}_N, \hat{\alpha}_N)$.

Now suppose we wish to compute $\hat{\alpha}_{N+1}$. We can write $p(\underline{I}_{N+1} \mid \alpha) = p(\underline{I}_N \mid \alpha) p(I_{N+1} \mid \alpha)$, so that upon using (20), there results

$$L_{N+1}(\alpha) \approx \left[\sum_{n=1}^{N} \ln p(I_n \mid \hat{\alpha}_N) \right] - \frac{1}{2}(\alpha - \hat{\alpha}_N)^T \Psi(\underline{I}_N, \hat{\alpha}_N)(\alpha - \hat{\alpha}_N) + \ln p(I_{N+1} \mid \alpha) \tag{22}$$

The appeal of (22) is that all the useful information in $\underline{I}_N$ is summarized in the quadric form, i.e., the second term on the right hand side of (22). Notice that only the two rightmost terms in (22) are functions of α. Now $\hat{\alpha}_{N+1}$ can be found approximately as the α that maximizes (22). Gradient descent can be used on (22). The gradient here is simply

$$\frac{-\partial L_{N+1}(\alpha)}{\partial \alpha} \approx \Psi(\underline{\bar{I}}_N, \hat{\alpha}_N)\cdot(\alpha - \hat{\alpha}_N) - \frac{\partial}{\partial \alpha}\ln p(I_{N+1}\,|\,\alpha)\,. \tag{23}$$

There is considerable computation here, since there are M^2 components for μ_1 in an M×M pixel patch, and α would therefore have $M^2 + K + 1$ components. A simplification is possible upon realizing that since the dependence of (22) on μ_1 is as a sum of two quadrics in μ_1, a simple explicit value can be found for $\hat{\mu}_{1(N+1)}$ in terms of $\hat{\alpha}_N$, I_{N+1}, $\Psi(\underline{\bar{I}}_N, \hat{\alpha}_N)$, σ^2, and $\mathbf{a}$. The resulting function to minimize is then a function of only σ^2 and $\mathbf{a}$, hence, only K+1 parameters. Gradient descent can be used for this purpose. This solution is explored in [9]. Though this should provide the most accurate estimate for α_T, for a number of reasons we have minimized a simpler function.

IV.C Approximate Likelihood Maximization

In this section, we treat $I_1(s)$, $s \in D$, as if it were $\mu_1(s)$. Then μ_1 is no longer treated as unknown -- only σ^2 and $\mathbf{a}$ are unknown. If our goal is to estimate $\mathbf{a}$ only, then we do not have to estimate σ^2 since σ^2 gives information only about the accuracy of the estimate for $\mathbf{a}$ (see [6]) but does not affect the value of the estimate for $\mathbf{a}$ (see [9]). Hence, $\alpha = \mathbf{a}$. For practical reasons, instead of letting D_n be an arbitrary subset in image plane n, we proceed analogously to the two image problems in Sec. III.A.. Hence, (17) becomes

$$p(I_n\,|\,\mathbf{a}) \approx (2\pi\sigma^2)^{-d_1/2}\exp\left\{\sum_{s\in D_1} -\frac{1}{2\sigma^2}\left[I_n(u_n(s,\mathbf{a})) - I_1(s)\right]^2\right\}. \tag{24}$$

Then,

$$L_{N+1}(\mathbf{a}) \approx \ln p(\underline{\bar{I}}_N\,|\,\hat{\mathbf{a}}_N, \sigma^2) - \frac{1}{2}(\mathbf{a} - \hat{\mathbf{a}}_N)^T\Psi(\underline{\bar{I}}_N, \hat{\mathbf{a}}_N, \sigma^2)(\mathbf{a} - \hat{\mathbf{a}}_N) + \ln p(I_{N+1}\,|\,\mathbf{a}, \sigma^2)\,. \tag{25}$$

Now, our goal is to compute $\hat{\mathbf{a}}_{N+1}$, the value of $\mathbf{a}$ that minimizes the negative of (25). We suggest a gradient descent algorithm similar to that used in Sec. III.A. Let $\hat{\mathbf{a}}_{N+1,k}$ denote the estimate for $\mathbf{a}_T$ after the k^{th} iteration in the $N+1^{st}$ stage (i.e., the $N+1^{st}$ stage is that following the input of the $N+1^{st}$ image and prior to the input of the $N+2^{nd}$ image). Then we compute $\hat{\mathbf{a}}_{N+1}$ as the limit of $\hat{\mathbf{a}}_{N+1,k}$ in (26).

$$\hat{\mathbf{a}}_{N+1,k+1} = \hat{\mathbf{a}}_{N+1,k} + \text{SCALE}\cdot\frac{\partial L_{N+1}(\mathbf{a})}{\partial \mathbf{a}}\bigg|_{\mathbf{a}=\hat{\mathbf{a}}_{N+1,k}}. \tag{26}$$

From (24) and (25),

$$\frac{\partial L_{N+1}(\mathbf{a})}{\partial \mathbf{a}} \approx \Psi(\underline{\bar{I}}_N, \hat{\mathbf{a}}_N, \sigma^2)(\mathbf{a} - \hat{\mathbf{a}}_N) + \sigma^{-2}\sum_{s\in D_1}\left[I_{N+1}(u_{N+1}(s,\mathbf{a})) - I_1(s)\right]\frac{\partial I_{N+1}(u_{N+1}(s,\mathbf{a}))}{\partial \mathbf{a}} \tag{27}$$

Once $\hat{\mathbf{a}}_{N+1}$ is computed, $\Psi(\underline{\bar{I}}_N, \hat{\mathbf{a}}_N, \sigma^2)$ can be updated to $\Psi(\underline{\bar{I}}_{N+1}, \hat{\mathbf{a}}_{N+1}, \sigma^2)$ by

$$\Psi(\underline{\bar{I}}_{N+1}, \hat{\mathbf{a}}_{N+1}, \sigma^2) = \Psi(\underline{\bar{I}}_N, \hat{\mathbf{a}}_N, \sigma^2) - \frac{\partial^2}{\partial \mathbf{a}\partial \mathbf{a}}\ln p(I_{N+1}\,|\,\mathbf{a}, \sigma^2)\bigg|_{\mathbf{a}=\hat{\mathbf{a}}_{N+1}} \tag{28}$$

with $-\frac{\partial^2}{\partial \mathbf{a}\partial \mathbf{a}}\ln p(I_{N+1}\,|\,\mathbf{a},\sigma^2)$ a matrix having ijth element

$$\sigma^{-2}\sum_{s\in D_1}\left\{\left[I_{N+1}(u_{N+1}(s,\hat{\mathbf{a}}_{N+1})) - I_1(s)\right]\frac{\partial^2 I_{N+1}(u_{N+1}(s,\hat{\mathbf{a}}_{N+1}))}{\partial a_j \partial a_i} + \left[\frac{\partial I_{N+1}(u_{N+1}(s,\hat{\mathbf{a}}_{N+1}))}{\partial a_j}\,\frac{\partial I_{N+1}(u_{N+1}(s,\hat{\mathbf{a}}_{N+1}))}{\partial a_i}\right]\right\}. \tag{29}$$

For brevity, denote $\Psi(\underline{\bar{I}}_N, \hat{\mathbf{a}}_N, \sigma^2)$ by Ψ_N. Then the incremental stereo algorithm is summarized as follows:

1. Read image 1.
2. Set $\Psi_1 = 0$.
3. For $N \geq 1$.
4. Read image N+1.
5. Compute $\hat{a}_N$ by using Eq. 26 iteratively until it converges.
6. Compute Ψ_{N+1} by Eq. 28.

IV.D Experiments With the Algorithm in Section IV.C

Figure 2 shows a sequence of nine computer generated images of a sphere. The images were generated by taking a few images of faces with a solid state T.V. camera, and using the computer to project these images onto a sphere. Using the pattern on the sphere generated in this way, the computer was then used to generate the images that should be seen by a camera at nine locations and with a specified CRF at each location. For this experiment the camera moved along a circular arc of radius 2000 units lying in a horizontal plane. The camera optical axis pointed to the center of this arc, and there was no rotation of the image plane about the optical axis. The angles between the camera optical axes in successive images were 5°. The patch of subimage used in each of the nine images is the region about the left eye of the rightmost face in the image. The patch of subimage is outlined as roughly a small square in white. The parameters specifying the sphere are $(x_0, y_0, z_0)^T$, the sphere center, and R, the sphere radius. In the experiments run, the sphere radius was assumed to be known and only the center was estimated. Table 1 shows the values of $\hat{a}_N$ found. The initial guess used for the sphere center was in error by a little more than the sphere radius of 128 units. The final estimate is in error by roughly two units. The optical axis of the camera moved through an angle of 40° from its first to its last position. These images were noiseless. However, some error is introduced because images are spatially quantized into pixels. Table 2 shows the estimates $\hat{a}_N$ for a more noisy image sequence. Each image here is the image in the corresponding position in Fig. 2 plus white Gaussian noise. The added noise has standard deviation of 5 units (i.e., variance of 25 units). The initial estimate $\hat{a}_1$ used here is also in error by about the sphere radius. The final error, based on nine images, is a little bit more than that in Table 1, but it is small. The accuracy of the algorithm appears to be remarkably good considering the small patches of data used in the estimation. In practice, image 1 would be partitioned into many squares, and a sequence of estimates would be obtained for each. The information obtained from each patch would be optimally combined using the methods presented in [3], thereby greatly improving the accuracy of the estimate of a_T. With the initial error used here, the algorithm in (26) went through about 8-10 iterations to compute $\hat{a}_N$ at each stage.

Figure 3 contains plots of $e_D(a)$, equation (30), as functions of x_0 and y_0, with z_0 held fixed at its true value, -2000.

$$e_D(a) = \sum_{n=1}^{N} \sum_{u \in D_n} \left[I_n(u) - \mu_1(s_n(u,a)) \right]^2, \tag{30}$$

The purpose of these plots is to show how $e_D(a)$, which is the function that must be minimized to maximize (19), narrows in the vicinity of its minimum as the number of images used increases. Since the height of $e_D(a)$, i.e., the distance between its minimum and maximum is an increasing function of the number of images used, we have only plotted the functions in the vicinity of their minima. That is, the plots stop at a height of roughly 3000 units above the minima. The functions shown are based on the use of 2, 6, and 10 images, respectively. It is seen that the functions narrow appreciably in going from the use of two images to the use of 10 images. In Fig. 4, curves of $e_D(a)$ are again shown , but only two images are used in each case. However,, the angle between the pair of camera optical axes varies, with angles of 1°, 5°, and 45° for the three plots shown. Notice how broad and flat the bottom of the curve associated with 1° is, whereas the curve associated with 45° is much narrower, as expected. However, it is still not as narrow as the curve in Fig. 3c where the angle between the optical axes of the first and tenth cameras is 40°. Hence, both the range of angles spanned and the number of images used is important. The other observation of interest is that the functions in Fig. 3 and those in Figs. 4a and 4b are smooth, whereas that in Fig. 4c is not. The multimodal benavior of Fig. 4c is due to the high frequencies in the pattern on the sphere surface. The effect is moderated when the angle between the optical axes of a pair of images is small, and the effect is also suppressed by the averaging that takes place when many more than two images are used.

V Incremental Stereo

A slightly different formulation is to write the joint likelihood for the image differences $I_2 - I_1$, $I_4 - I_3, \ldots, I_N - I_{N-1}$. The joint likelihood can be written as

$$\prod_{n=1}^{N/2} \left[2\pi(2\sigma^2)\right]^{-d_{2n-1}/2} \exp\left\{-\frac{1}{4\sigma^2} \sum_{u \in D_{2n-1}} \left[I_{2n}(u_{2n}(u,a)) - I_{2n-1}(u)\right]^2\right\}. \tag{31}$$

Here, $u_{2n}(u,a)$ in $I_{2n}(u_{2n}(u,a))$ denotes the point in D_{2n} that the point u in D_{2n-1} maps to. The mean value functions $\mu_1(s_n(u,a))$ do not appear here since the expectation of $I_{2n}(u_{2n}(u,a_T)) - I_{2n-1}(u)$ for each u is 0. Also, the variance of this difference for each u is $2\sigma^2$. Then $a_{N+2}^{\uparrow}$ is to be chosen to minimize

$$-L_{N+2}^{\uparrow}(a) = \sum_{n=1}^{(N+2)/2} \frac{d_{2n-1}}{2} \ln(4\pi\sigma^2) + \sum_{n=1}^{(N+2)/2} \frac{1}{4\sigma^2} \sum_{u \in D_{2n-1}} \left[I_{2n}(u_{2n}(u,a)) - I_{2n-1}(u)\right]^2. \tag{32}$$

Again, it is computationally undesirable to store the N+2 images and also to process all of them simultaneously in order to compute $a_{N+2}^{\uparrow}$. Hence, as in Sec. IV.C., we use an asymptotic approximation, Gaussian in a, to represent (31) when computing $a_{N+2}^{\uparrow}$.

Table 3 contains the estimates $a_N^{\uparrow}$ based on a sequence of images including those in Fig. 2. Note that the angle between the optical axes for the first and last camera positions used for the images in Fig. 2 is 40°. The viewing angle spanned by for the 18 camera positions used in computing Table 3 is 85°. Notice that even with using 18 images---9 pairs of differences---the algorithm in Sec. IV.C is considerably more accurate. The reason is that the algorithm minimizing (32) uses only the differences in pairs of images taken with camera optical axes that are almost parallel. Hence, it is small baseline stereo and suffers many of the disadvantages of the use of optical flow. If the images are noisy, the relative accuracy of this algorithm would probably degrade considerably. It is interesting to note that the size of the angle between the optical axes of the first and last images is not very important here. Rather, improved accuracy comes from using many pairs of images in order to average out the effects of noisy perturbations.

On the other hand, small angle stereo permits computational advantages which we briefly touch upon. If the camera does not move much in going from the (2n-1)th to the (2n)th position, $u_{2n}(u,a)$, where $u \in D_{2n-1}$, is close to u since CRF(2n) is close to CRF(2n-1). Hence, we can use the Taylor series expansion:

$$I_{2n}(u_{2n}(u,a)) \approx I_{2n}(u) + \left[\frac{\partial I_{2n}(u)}{\partial u}\right]\left[\frac{\partial u_{2n}(u,a)}{\partial b}\right] \Delta b. \tag{33}$$

Thus,

$$\left[I_{2n}(u_{2n}(u,a)) - I_{2n-1}(u)\right]^2 \approx \left\{\left[I_{2n}(u) - I_{2n-1}(u)\right] + \left[\frac{\partial I_{2n}(u)}{\partial u}\right]\left[\frac{\partial u_{2n}(u,a)}{\partial b}\right] \Delta b\right\}^2, \tag{34}$$

where Δb is an incremental vector specifying the incremental rotation and the incremental origin translation for CRF(2n) in term of CRF(2n-1). The desirability of the approximation is that in minimizing (32) with respect to a it is no longer necessary to compute the $u_{2n}(u,a)$ and then the arrays $I_{2n}(v,a)$ and $\left.\frac{\partial I_{2n}(v)}{\partial v}\right|_{v=u_{2n}(u,a)}$ for all $u \in D_{2n-1}$. Rather we can just use the arrays $I_{2n}(u)$ and $\frac{\partial I_{2n}(u)}{\partial u}$, $u \in D_{2n-1}$, directly. This makes for a considerable reduction in required computation. Furthermore, note that when computing the gradient of (34) with respect to a, only the term $\frac{\partial u_{2n}(u,a)}{\partial b}$ is a function of a, and this function is very simple as seen in Eq. 6a.

The final remark of interest is that for the planar surface described in the appendix, the use of Eqs. 6, 34, and A2 (from the appendix) in Eq. 32 permits a simple *explicit* solution for $a_{N+1}^{\uparrow}$, the value of a that minimizes (32).

VI Conclusion

In this paper, for the first time the joint likelihood of two or more images as a function of the a priori unknown 3-D surface to be estimated is derived. This permits the full range of Bayesian analysis, estimation, and recognition techniques to be applied to the 3-D surface inferencing problem. In particular, in this paper we develop a recursive

algorithm for the maximum likelihood estimation of a parameterized surface based on a sequence of images taken, perhaps, by a moving camera. This recursive estimator should be significantly more accurate than the use of the extended Kalman filter, since the latter uses a linearization about the N^{th} stage estimate to compute the $N+1^{st}$ stage estimate whereas we use the complete information in the $N+1^{st}$ image.

APPENDIX: The Plane

We derive the expression for the vector $\partial z/\partial a$ for a plane. Note that there are a number of different sets of parameters that can be used for representing a plane (or a cylinder, or a more general surface). We use the canonical parameterization in this section. We use the equation

$$0 = g(x,y,z) = \beta_1 x + \beta_2 y + \beta_3 z - d \tag{A1}$$

subject to the constraint

$$0 = f(x,y,z) = \beta_1^2 + \beta_2^2 + \beta_3^2 - 1 \tag{A1a}$$

Note, $|d|$ is the distance from the plane to the origin in this representation. It is assumed that the plane is in general position, because if, e.g., $\beta_3 = 0$, then the plane normal is orthogonal to the first camera's optical axis, and the plane surface is not seen by the first camera since the camera then sees only the plane's edge. Eq. (A1a) can be used to solve for β_3 in terms of β_1 and β_2. Hence we can take a to be $a^T = (\beta_1,\beta_2,d)$. Now $\frac{\partial z}{\partial a} = -\frac{\partial g/\partial a}{\partial g/\partial z}$. Using (A1a), we get $\frac{\partial \beta_3}{\partial \beta_1} = -\frac{\partial f/\partial \beta_1}{\partial f/\partial \beta_3} = \frac{-2\beta_1}{2\beta_3} = -\frac{\beta_1}{\beta_3}$. Similarly, $\frac{\partial \beta_3}{\partial \beta_2} = -\frac{\partial f/\partial \beta_2}{\partial f/\partial \beta_3} = -\frac{\beta_2}{\beta_3}$. Hence, $\frac{\partial g}{\partial z} = \beta_3$

$$\frac{\partial g}{\partial a} = \begin{cases} \frac{\partial g}{\partial \beta_1} = x + z\frac{\partial \beta_3}{\partial \beta_1} = x - z\frac{\beta_1}{\beta_3} = \frac{\beta_3 x - \beta_1 z}{\beta_3} \\ \frac{\partial g}{\partial \beta_2} = y + z\frac{\partial \beta_3}{\partial \beta_2} = \frac{\beta_3 y - \beta_2 z}{\beta_3} \\ \frac{\partial g}{\partial d} = -1 \end{cases}$$

Thus,

$$\frac{\partial z}{\partial a} = \left[\frac{\beta_1 z - \beta_3 x}{\beta_3^2}, \frac{\beta_2 z - \beta_3 y}{\beta_3^2}, \frac{1}{\beta_3}\right]. \tag{A2}$$

ACKNOWLEDGEMENT

Partial support was provided by NSF Grant #ECS-81-19676, ARO Grant #DAA-LO3-86-K-0011, the IBM Co., and by the University of Buenos Aires.

REFERENCES

[1] D.H. Ballard, C.M. Brown, Computer Vision, Prentice Hall, Englewood Cliffs, N. J., 1982.

[2] P.J. Besl, R.C. Jain, "Invariant Surface Characteristics for 3D Object Reconstruction in Range Images," Computer Vision, Graphics, and Image Processing, 33, 1986, pp 33-80.

[3] R.M. Bolle, D.B. Cooper, "On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data," IEEE Trans. on PAMI, Sept. 1986, pp 619-638.

[4] B. Cernuschi-Frias, P.N. Belhumeur, D.B. Cooper, "Estimating and Recognizing Parameterized 3-D Objects Using a Moving Camera," Proc. IEEE Comp. Soc. Conf. On Computer Vision And Pattern Recognition, San Francisco, June 1985, pp 167-171.

[5] B. Cernuschi-Frias, D.B. Cooper, F.G. Amblard, "Estimation by Multiple Views of Outdoor Terrain Modeled by Stochastic Processes," to appear in Proc. SPIE Cambridge Symposium on Optical and Optoelectronic Engineering: Intelligent Robots and Computer Vision, Cambridge, Mass., Oct. 28-31, 1986.

[6] B. Cernuschi-Frias, D.B. Cooper, P.N. Belhumeur, Y.P. Hung, "Toward A Bayesian Theory for Estimating and Recognizing Parameterized 3-D Objects Using Two or More Images Taken From Different Positions," under review for journal publication. Also, Brown University, Division Of Engineering, LEMS Tech Report #32, December, 1986.

[7] D. Duda, P. Hart, Pattern Classification And Scene Analysis, John Wiley, 1973, p. 381.

[8] O.D. Faugeras, N. Ayache, B. Faverjon, "Building Visual Maps by Combining Noisy Stereo Measurements," Prc. 1986 IEEE Conf. On Robotics And Automation, San Francisco, April 7-10, 1986, pp 1433-1438.

[9] Y.P. Hung, "Maximum Likelihood Estimation of Parameterized Surfaces In 3-D space Using A Sequence of Images," M.Sc. thesis, 1987, Brown University, Division of Engineering.

[10] Y. Ohta, T. Kanade, "Stereo by Intra-and Inter-Scanline Search Using Dynamic Programming," IEEE Trans. on PAMI, March 1985, pp 139-154.

[11] A.M. Waxman, K. Wohn, "Image Flow Theory: A Framework For 3-D Inference From Time-Varying Imagery," a chapter to appear in Advances In Computer Vision, ed. C. Brown (Erlbaum Publishers).

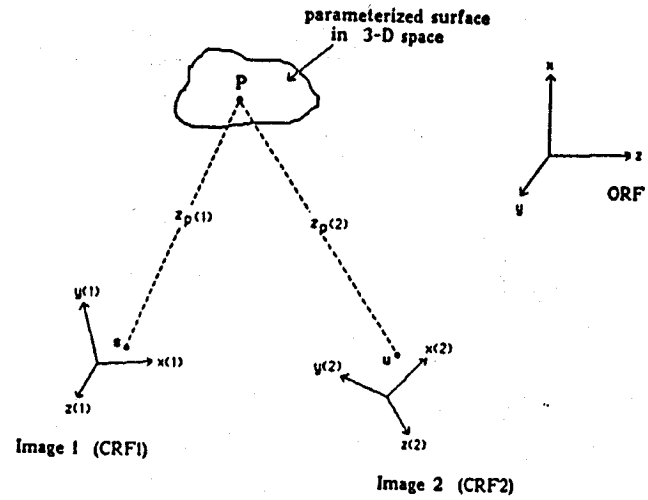


Figure 1a

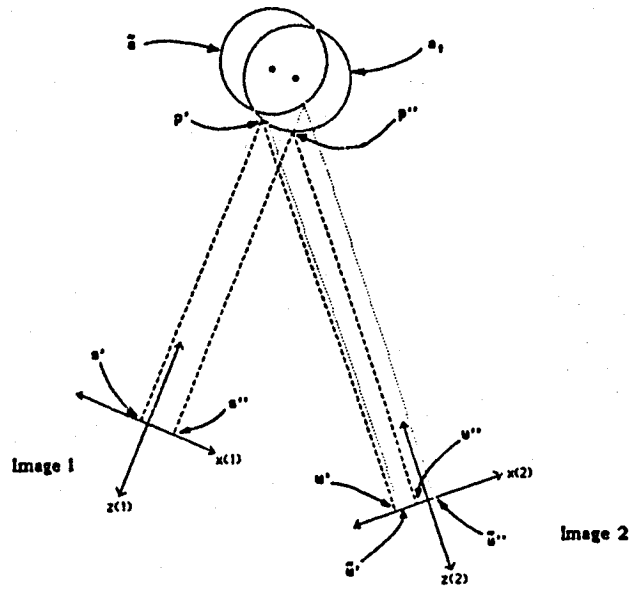


Figure 1b

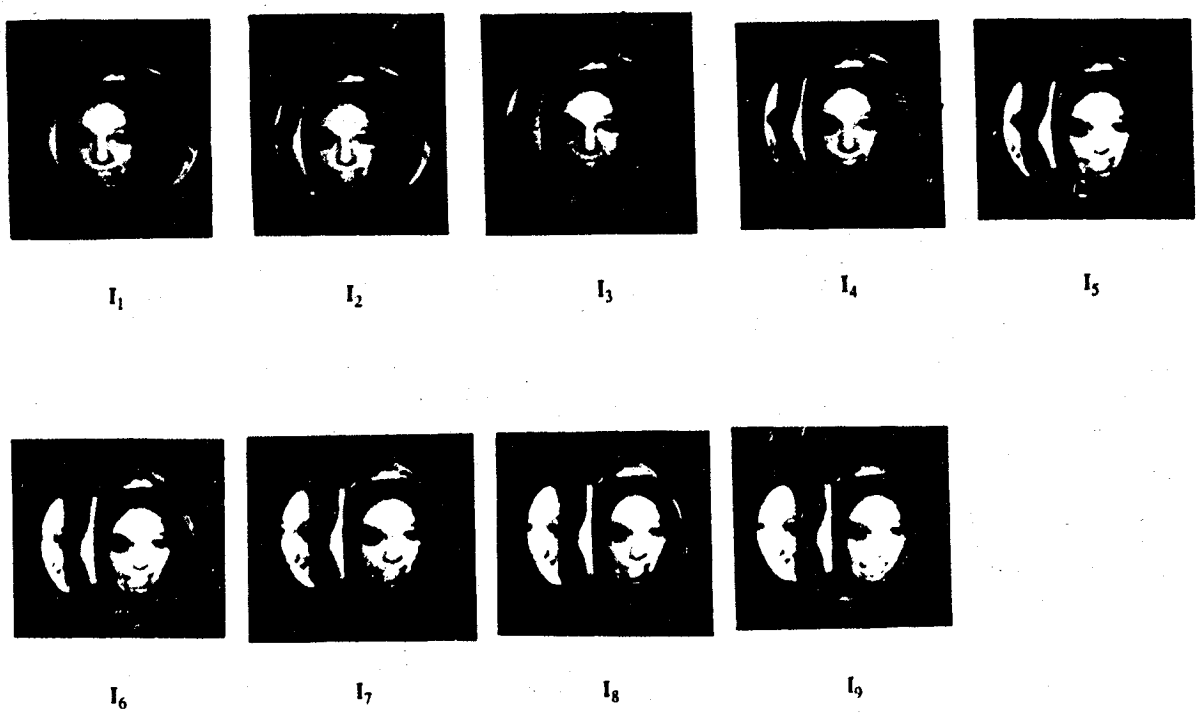


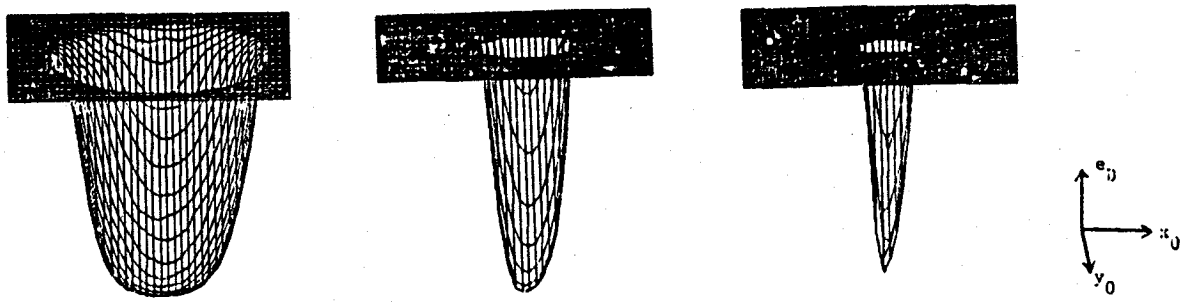
Figure 2

# of image	$\hat{\mathbf{a}}_N$			
	x_0	y_0	z_0	R
initial $\hat{\mathbf{a}}_1$	60.0	60.0	-2120.0	128
2	16.9	9.7	-2001.4	128
3	11.4	6.9	-2002.3	128
4	7.4	4.1	-2001.2	128
5	4.3	2.9	-2000.8	128
6	3.1	2.1	-2000.5	128
7	1.7	1.9	-2000.1	128
8	1.8	1.8	-2000.2	128
9	-0.6	1.9	-2000.1	128
true $\mathbf{a}$	0.0	0.0	-2000.0	128

Table 1

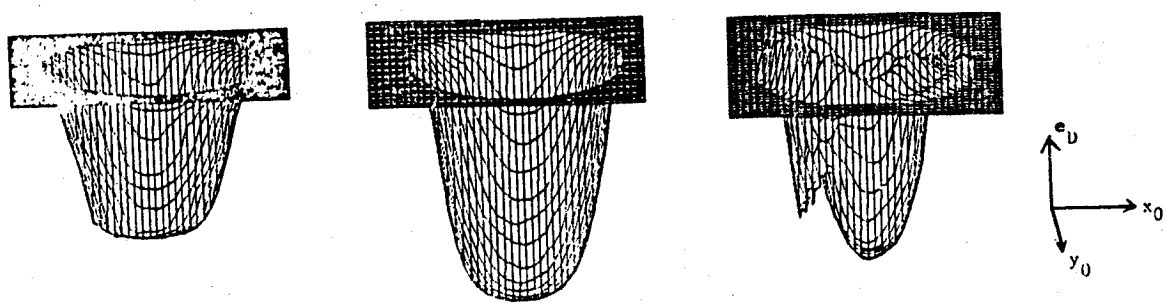
# of image	$\hat{\mathbf{a}}_N$			
	x_0	y_0	z_0	R
initial $\hat{\mathbf{a}}_1$	50.0	50.0	-2080.0	128
2	11.0	6.9	-1993.0	128
3	2.7	-1.6	-1993.6	128
4	-0.6	-13.4	-2000.1	128
5	0.7	-1.7	-2001.9	128
6	0.8	8.3	-2002.1	128
7	1.1	1.1	-2001.8	128
8	0.6	1.6	-2001.5	128
9	1.0	1.5	-2001.3	128
true $\mathbf{a}$	0.0	0.0	-2000.0	128

Table 2



(a) based on I_1, I_2 (b) based on $I_1, I_2, \ldots, I_6$ (c) based on $I_1, I_2, \ldots, I_{10}$.

Fig. 3 Error function



(a) with 1° between optical axes (b) with 5° between optical axes (c) with 45° between optical axes

Fig. 4 Error function based on two images

number of images used	$\hat{\mathbf{a}}$			
	x_0	y_0	z_0	R
initial $\hat{\mathbf{a}}$	50.0	50.0	-2086.0	128
2	26.8	20.3	-2001.8	128
4	16.9	13.3	-2001.1	128
6	10.1	10.4	-2001.4	128
8	8.6	8.8	-2001.2	128
10	10.3	8.9	-1999.9	128
12	5.3	4.9	-2002.1	128
14	5.5	1.3	-2000.4	128
16	5.9	-1.4	-2001.0	128
18	-0.8	-2.2	-2001.9	128
true $\mathbf{a}$	0.0	0.0	-2000.0	128

Table 3